Amendments to the Specification

Please replace the paragraph at page 18, line 25 through page 19, line 2 with the following amended paragraph:

The hosts 12 typically accept queries that are requests for data stored on mass storage devices, such as hard disk drives 23. The requests may originate from any number of applications, typically business intelligence applications, that may be residing on local processors 21 or client computers 36 or separately running application software [[30]] 39, that may originate through a computer network 33 or locally. Queries are typically provided in a format such as Structured Query Language (SQL), Open DataBase Connectivity (ODBC), Java DataBase Connectivity (JDBC), or the like.

Please replace the paragraph at page 20 lines 10-18 with the following amended paragraph:

In some embodiments, JPU memory 27 is relatively smaller than host 12 memory. Thus, processor memory is a precious resource. As such the present invention advantageously provides (i) data processing of record data in a continuum or streams, and (ii) a flow of data (or pipeline or overall logical data path) from storage disk 23 to PSDP 28, to JPU memory 27, to JPU CPU 26, to internal network 34, to host 12 memory, to host CPU, to an output buffer (to enduser/client computers or applications 21, 36, [[30]] 39). Such streaming of record data and data flow/pipeline provide improved data processing heretofore unachieved by the prior art as will become apparent by the following description.

Please replace the paragraph at page 22 lines 2-10 with the following amended paragraph:

The system architecture exhibits further aspects of asymmetry in that one or more so-called Large Job Processing Units (LJPUs) 30 can also play a part in processing queries. Each LJPU [[22]] 30 consists of a network interface for receiving job requests and delivering replies, and one or more general purpose Central Processing Units (CPUs) ~~32-1, …, 32-p~~ 132-1,...,132-p (each of which may have their own internal memory), as well as a shared memory [[38]] 138. The CPUs [[32]] 132 in the LJPUs 30 preferably represent a relatively powerful computing resources, consisting of a relatively high speed processor that has access to relatively large

amounts of memory. The LJPUs may be organized as an SMP that share portions of memory [[38]] 138.

Please replace the paragraph at page 22, lines 19-23 with the following amended paragraph:

LJPU(s) [[20]] 30 may also typically use a multi-tasking Operating System (OS) to allow receiving, processing, and reporting the results from multiple jobs in a job queue. In the preferred embodiment, the OS should also support overlapping job execution. To coordinate this, the OS typically is responsible for scheduling and prioritizing requests according to a number of factors that are determined in real time.

Please replace the paragraph at page 36, lines 5-6 with the following amended paragraph:

LJPU Scheduler [[374]] 377

- Schedules jobs to run on the LJPU

Please replace the paragraph at page 41, lines 21-29 with the following amended paragraph:

The terms "flow through" and "filtered" are used to differentiate DMA reads. In flow-through mode, also referred to as raw read mode, data moves directly from the input to the output of the data engine 400 without being filtered. Data that is filtered has been processed, perhaps by culling records via the comparison and/or transaction ID circuits, but certainly by reformatting the records into tuple format, during which uninteresting fields can be dropped and PSDP-generated fields added. The processing of culling records is called the "restrict". The process of formatting fields into tuples is called the "project" (pronounced, as in ~~as in~~ "throwing" something.)

Please replace the paragraph at page 49, line 28 - page 50, line 6 with the following amended paragraph:

In this example, with reference to Figs. 1 and 3, the query is passed from a user 21, 36, [[30]] 39 (say Intelligence Applications [[30]] 39 for this example) over the external network 33 to the host 12. On the host 12, Postmaster/Postgres 201, 202 and Plan Generator 204 respond to

the query by parsing it and creating tentative execution plans. The Plan Generator 204 takes into consideration the unit of input and output (i.e., streams of records) of the operators and generates record processing plans accordingly such as to avoid intermediate materialization. Techniques of U.S. Provisional Patent Application 60/485,638 for "Optimized SQL Code Generator II" previously referenced may be used. The tentative execution plans not only specify the above job description, but also may specify whether jobs can run concurrently or must run in sequence on the JPUs 22. The Plan Optimizer 205 selects one of the plans and optimizes that plan and passes it to the Plan Link. The Plan Link 206 expands the plan as necessary, based on where parts of the plan will be executed, and then passes the expanded plan to the Host Dispatch 208. The Host Dispatch 208 sends individual jobs within the plan to the respective locales (JPUs 22) for execution. In this example, jobs 1-6 are sent to the JPU 22 for execution with job 7 reserved for host 12.

Please replace the paragraph at page 50, lines 13-23 with the following amended paragraph:

The Host Dispatch 208 may thus combine Jobs 3-6 into one streaming job because they can all be implemented in a streaming manner without materialization of intermediate result sets. This combined job scans the SalesDetail table, with its restrictions and projections. As the tuples are received from the scan run by the PSDPs 28, each tuple is joined with TEMPStore and TEMPCustomer and aggregated. On the aggregation node, as each new customer ID is received, the previous one and its sums are sent to the host 12, where Job 7 is then invoked in a streaming fashion, to return the aggregated tuples (subsequently formatted into records) through the ODBC connection 38 back to the user [[30]] 39. Materialization is thus delayed from Jobs 4 and 5 and performed in Job 6 just before returning the results of aggregating to the host 12 autonomous/asynchronous.

Please replace the paragraph at page 51, line 26 - page 52, line 5 with the following amended paragraph:

JPU 22c processing Job 4 receives the broadcast stream of record data from JPU 22b and uses the same as input into the first operation of Job 4 (JOIN WITH TEMPStore) and likewise

processes this received stream of record data. In this way, there is a flow of record data on a logical data path 700 as prescribed by the Jobs 1-7 being processed within nodes 22, 12 and across nodes 22, 12. In particular, streams of record data follow a data flow pipeline defined by the sequence of job operators within nodes 22, 12 and across nodes 22, 12 of the system network. The pipeline 700 extends from disk 23 to JPU 22 memory, to internal network 34, to host 12 memory, to ODBC connection 38 or other connection to the end user requester 21, 36, [[30]] 39.

Please replace the paragraph at page 54, lines 7-14 with the following amended paragraph:

The preferred embodiment utilizes asymmetric scheduling whereby each JPU22 is able to schedule jobs for itself without regard to how jobs are scheduled for other JPU's. This allows each JPU to complete its assigned tasks independently of other JPU's thereby freeing it to perform other tasks. Where many requesters (users or applications) 21, 36, [[30]] 39 are making multiple requests on multiple databases at substantially the same time, it is understood that the job queue within a given JPU will quickly become filled with different instructions/operations to perform. By making the JPU operations asynchronous the overall throughput is greatly increased here.

Please replace the paragraph at page 54, lines 15-24 with the following amended paragraph:

The job listener component 210 in the host 12 first coordinates job responses from multiple JPUs 22. In particular, the job listener 210 waits to receive results data from each JPU before reporting back to the Host/Event Handler 252 that a particular job has been completed. To expedite this, each job can be tagged with a unique job identifier (JID). When each JPU returns results from its respected aspect of a job, the JID is included, as well as an identifier for the JPU. The Host Event Hander 252 thus knowing how many JPUs 22 are active can then tally responses from the JPUs to ensure that job identifiers are received from each before taking the next step in a plan that has jobs that must be run sequentially and before reporting results back to the requesting user/application 21, 36, [[30]] 39.

Amendments to the Drawings

The subject application was filed with eight (8) sheets of drawings bearing Figs. 1 - 7. Eight sheets of replacement drawings consisting of Figs. 1 - 7 corrected to be in compliance with C.F.R. § 1.84 and CFR § 1.121 are attached.

In particular, in Fig. 1 Local App. 28 is being amended to recite reference number 21 as set forth on Specification page 18, line 28 as originally filed. Memory 33 is being amended to be referenced as memory 138 to correct duplicate use of the reference number 33 with external network 33. Processors (CPU's) 32-1 through 32-p are being amended to be referenced as processors 132-1 through 132-p, to correct duplicate use of reference number 32 with TCP/IP network connection 32. Further reference numbers 32, 34 and 38 are being added to Fig. 1. Support for addition of these reference numbers is found at least on Specification page 23, line 16, page 24, line 3, page 52 lines 3-5 and in Fig. 7 as originally filed.

In Fig. 3 the Cube Builder user interface reference number 242 was previously omitted, and is now being inserted. Support for this correction is found at least on Specification page 28, line 20 as originally filed.

Host Disk Manager 250 of Fig. 3 is being amended to read Host Disk Manager 215 to correct duplicate use of reference number 250 with JPU downloader 250. Support for this amendment is found at least on Specification page 28, line 24 and page 29, line 3 as originally filed.

In Fig. 4A, reference number 22 is repositioned to more clearly indicate the JPU software component.

In Fig. 4B, the reference number 30 for LJPU was previously omitted and is now inserted. Support for this amendment is found at least on Specification page 34, line 6 and Fig. 1 as originally filed.

Further, the lead line for reference number 350 is being corrected to more accurately indicate the communication layer (left hand side) of Fig. 4B. Support for this amendment is found at least on Specification page 34, line 14 as originally filed.

LJPU scheduler 374 of Fig. 4B is being amended to recited reference number 377 to correct duplicate use of reference 374 with LJPU Boot/Init 374. Support for this amendment is found at least on page 35, line 22 of the specification as originally filed.

In Fig. 7, reference number 30 is being corrected to recite requester (e.g., application) 39. Support for this amendment is found at least in Fig. 1 as originally filed.

Copies of the amended figures (Figs. 1, 3, 4A, 4B and 7) as originally filed but marked up to show the foregoing amendments are also attached.

Attachment: Replacement Sheet
Annotated Marked-Up Drawings